Fall 2021—Assignment 3

Supervised Learning

Using the provided dataset, implement 3 different classification algorithms (k-nearest neighbor, perceptron, and decision tree) to determine whether a candidate gets the majority of votes. For this assignment, you will use python3 and (optionally) the numpy library for vector/matrix-based scientific computation in python.

The assignment file you will submit is assignment3.py. Please complete the methods within the `class` definitions provided. You are free to add additional helper methods and classes. All the code will be run using the run_assignment3.py file. You can modify this file for your own testing, but you can **only upload the assignment3.py file** to Brightspace, so make sure all your code is in that file. You can expect that any variable in the "run" file could be changed during evaluation though its data type will remain the same. For example, the shape of the data could change (i.e. there may be a different number of points or features). If you **don't hardcode values**, you should be fine.

During evaluation, we will let the "run" file run for up to 1 minute, but you should really aim for under 10 seconds total (assuming an average modern laptop). Using numpy arrays is not required, but it is recommended. Numpy provides many vector operations that you can run over data (often replacing costly for-loops with optimized one-line operations). These can help keep your code clean and simple while massively improving performance. The numpy documentation may be a helpful reference.

Your code should not print anything to the console when you submit your assignment.

Data

The attached csv file contains all the data. The run file handles importing it and converting it to numpy arrays. A description of the dataset is in the run file.

Algorithms

K-Nearest Neighbor

Use Euclidean distance between the features. Choose a k value and use majority voting to determine the class. The k value is provided to the knn class. Please implement the methods `train` (which simply stores the data) and `predict` (which runs the KNN algorithm). The `distance` function is provided for you, and you can assume that all data is continuous. Ties can be broken arbitrarily by picking either class.

Perceptron

For the perceptron, multiply the inputs by a weight matrix and then pass the output through a single heaviside function (a.k.a. step function) to get the output. Don't forget the bias. Train the perceptron using the perceptron learning algorithm. The weight matrix and bias are initialized in the run file to facilitate grading. You must update the weights, but don't change the shape or type of the numpy array. The number of steps is defined in the run file. For each step, update the model on a single datapoint.

Decision Tree

Use the ID3 algorithm as defined in the slides. Since all the data is continuous, they have to be broken down into bins. This has been done for you in the `preprocess` method. You can preprocess the data in the train and predict methods (which you will implement) before evaluating. The `preprocess` functions break the data into bins of equal width along each feature dimension. If there are ties in selecting the majority class or the maximum information gain, pick one.